NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-761

Nondeterministic Data Base for Computerized Visual Perception

Y. Yakimovsky

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

February 1, 1976

PREFACE

The work described in this report was performed by the Space Sciences Division of the Jet Propulsion Laboratory.

CONTENTS

PRECEDING PAGE BLANK NOT FILMED

ABSTRACT

A description is given of the knowledge representation data base in the perception subsystem of the Mars robot vehicle prototype being implemented at JPL. Two types of information are stored. The first is generic information that represents general rules that are conformed to by structures in the expected environments. The second kind of information is a specific description of a structure, i. e., the properties and relations of objects in the specific case being analyzed. This paper 's limited in scope to the description of the syntax and semantics of the data structure. The generic knowledge is represented so that it can be applied to extract and infer the description of specific structures. The use of the generic model in the inference process is only briefly described where needed to justify the generic knowledge representation, and it will be thoroughly described in a following publication.

The generic model of the rules is substantially a Bayesian representation of the statistics of the environment, which means it is geared to representation of nondeterministic rules relating properties of, and relations between, objects. The description of a specific structure is also nondeterministic in the sense that all properties and relations may take a range of values with an associated probability distribution.

Introduction

There were a few attempts to build useful models of generic information. The traditional method of representation is the use of a set of deterministic axioms expressed in mathematical logic (Ref. 1, Chaps. 6, 7, 8). This approa·h was used successfully in generation and analysis of mathematical and physical models and theories. Unfortunately, attempts to use this method of representation by computer programs for automatic analysis ran into the problems of computational impracticability as was foreseeable from complexity theory (Ref. 2). In addition to the inherent computational impracticability of the use of most nontrivial axiom systems, most rules in the real world are nondeterministic. That is, for almost all rules, exceptions can be found.

Following the realization that logic-based (deterministic) rule systems are impractical, attempts to design practical nondeterministic inference systems were made (Refs. 3, 4).

This article represents our approach to the representation of nondeterministic generic rules and to nondeterministic representation of the specific cases being analyzed.

Environment Description — Specific Structure Representation

In the abstract, a specific structure can be represented as a set of objects, properties of those objects and relations between the objects. Our domain of specializations is the representation and the analysis of pictorial information. In that domain, typically there are the following classes of objects: (1) scenes (picture frames), (2) three-dimensional bodies, (3) two-dimensional regions (pictorial images of three-dimensional surfaces), (4) one-dimensional lines, (5) one-dimensional boundaries between regions, (6) vertices, etc.

In each specific instance of the environment (for example, a specific picture frame), there appear a few objects of those types. Each one of these objects has properties and is related to other objects. Figure 1 gives a partial data structure describing an image.

The knowledge data base is implemented in SAIL, an Algol-based associative data base. The unfamiliar reader is referred to Refs. 5 and 6 for the semantic definition of the data type.

Objects and Classes: The list of possible "classes" of objects is predetermined by the repertoire of the generic model. But when representing an instance structure, each "class" is a "set" data structure. Such a "set" is a data structure containing an unsorted list of nonrecurring pointers to the data structures representing the known objects of that class in the specific structure. For instance, the set associated with the class "regions" when representing the model in Fig. 1 will contain pointers to regions R_1, R_2, R_3, and R_4. The object data structure itself is an item containing the object's name (for man-machine interaction), the name of its class a some basic preprogrammed properties. Usually each object in a structure is in relation "part of" with a global object (the "scene" in the visual pictures case). The global object allows the definition of global properties like camera position and lighting condition which affects the analysis of all substructures.

Representation of Properties: The generic model of the environment defines a set of features which are functions that act on elements of a specified class and range into the integer domain. These functions are called features. For example, the following are some of the features of the objects of class regions: "area," "color," "average light intensity," "shape."

Features which act on objects of type boundary include "length," "difference in light intensity across the boundary line," etc.

The value of a feature when applied to an object is a property of the object. For instance, application of the feature "area" to region R_1 may result in the value 7, which is a property of R_1.

The representation of properties is made a bit more complicated, because measurements may be unreliable and take a range of possible values. For instance, estimating the "label" (meaning) of the body imaged on R_1 may be speculative. To represent ambiguity, properties are allowed to take range of values with an associated probability estimate of the validity of that value.

The association of feature, object, and property is represented by the associative data structure of SAIL, and may look like this:

$$\text{Area} \otimes R_3 \equiv \begin{pmatrix} 16 & 0.05 \\ 15 & 0.9 \\ 14 & 0.05 \end{pmatrix}$$

This will mean that the current estimate is that the area of R_3 is 16 with probability 0.05, 15 with probability 0.9, and 14 with probability 0.05.

The properties data structure is further complicated by two factors. First, a property may change in time; hence, some estimate on the period of validity of that property must be given. Secondly, often the reliability of the estimate of a property depends on the resources spent in measurement and analysis. Hence, an indication of the amount of resources (compute time) allocated to obtaining the specific property, the amount of resources actually used to obtain that estimate and the real time (when) the estimate was obtained is associated with the feature. As a result, the property data structure (the item's content) contains the following kind of information:

(1) Valid-from date

(2) Valid-till date

(3) Obtained-at date

(4) Resources allocated to obtain the property's estimate

(5) Resources used to obtain the estimate

(6) Estimate value list

$$\begin{pmatrix} val_1 & P_1 \\ val_2 & P_2 \\ \vdots & \vdots \\ val_n & P_n \end{pmatrix}$$

(more compact representation of the probability distribution of values is used when practical)

All dates are measured in absolute time, and if a property is assumed constant in the effective time, the validity dates may be set to $-\infty$ or $+\infty$ to indicate permanency in one or two time directions in the scope of the analysis.

Resources are measured in the number of machine cycles which were used or were allocated to be used to obtain the estimated property.

There can be more than one property associated with a pair of a feature and a specific object. For instance, the color of an object may be blue from $06\underline{^{00}}$ to $17\underline{^{35}}$, red from $17\underline{^{35}}$ to $18\underline{^{30}}$, and black otherwise in the period of interest. The associative storage mechanism SAIL provides for that type of multiplicity of associations of few properties with a single pair of a feature and an object.

Some feature values (properties) and relation values (object lists) are being put in manually as training samples for a learning process where the generic model is being improved or analyzed. Such properties are marked as "special" knowledge for training purposes. The learning process is not described at all within the scope of this article, and no further reference to that point will be made.

Representation of Values of Relations (Objects List)

A number of relations are defined for each class of objects in the generic model. The value of a relation applied to an object of the appropriate class will be a list of objects all of one class (possibly different class than the class of the object operated on) which satisfy the relation. For instance, "boundary of" will be a relation which operates on object of class "regions" and will come back with a list of objects of class "boundary lines" which are boundaries of that region.

The limitation that relation applies to one object at the time was imposed to simplify implementation.

In general, relations are n-ary, not binary, as in the present implementation; that is, the relation is between more than two objects. The limitation may be bypassed by defining "combination" objects; that is, define an object class whose elements stand for an order list of simpler objects. The current system provides for composite objects and guarantee uniqueness in their representation.

Typically, application of relation to an object results in the following association being put in the data base:

$$\text{Relation} \otimes \text{object}_0 \equiv \begin{Bmatrix} \text{object}_1 & W_1 \\ \text{object}_2 & W_2 \\ \text{object}_3 & W_3 \\ \vdots & \\ \text{object}_n & W_n \end{Bmatrix}$$

The relation is the name as defined in the generic model. Object_0 is an instance of an object of the class of objects upon which the relation works. Each one of object_i $(1 \le i \le N)$ is estimated to satisfy the relations with the corresponding probability W_i. In fact, the relations values defines fuzzy sets (Ref. 7). All of these objects belong to the same class of objects which is the range of the relation. Clearly, N varies with different applications of the relation. W_i must be higher than 0.5; otherwise, the corresponding object is not included in the list to save storage. Note, that the W_i's do not have to add up to one. There may be cases where more than one object satisfies the relation, and there may be cases where the list is empty; that is, no object appears to satisfy the relation.

As is the case with properties, the "list" of objects which satisfy a certain relation is time and resource dependent. The more the resources are expanded for searching for such objects, the more of them are likely to be found. Similarly, the period of time during which the relation will be valid may be limited. As a result, the data structure of the object list contains also the following information:

(1) Start of validity period (date)

(2) End of validity period (date)

(3) Approximated time when list was computed

(4) Resources (compute time) allocated to obtaining the list

(5) Resources used in obtaining the list

There are three types of evaluations of relations. The first corresponds to the existential quantifier ($\exists$) in logic. This existential search tries to find an object which satisfies the relation with high enough probability as specified in the call, and it terminates successfully as soon as one is found. The second type of evaluation is an exhaustive search. Its aim is to find as many objects which satisfy the relation with high probability within the bound of the resources allocated. The third call operates an exhaustive search and generates a full objects list, but it returns as value only the one object which maximizes the "belong" property among all found objects. The object list contains also a mark as to which type of search was used to obtain it.

Most relations in the generic model are related to two special features. The first feature is a filter feature for the relation. It operated on a composite object of two elementary objects (object_0, object_1) and it takes only two values, 0 or 1. It takes the value 1 if object_1 satisfies the corresponding relation to object_0.

Example:

$$\text{Filter}_c \otimes \left\{\text{Object}_0, \text{Object}_1\right\} \equiv \left\{\begin{matrix} 0 & 0.4 \\ 1 & 0.6 \end{matrix}\right\}$$

is equivalent to

$$\text{Relation}_c \otimes \text{Object}_0 \equiv \left\{\begin{matrix} \vdots & \\ \text{Object}_1 & 0.6 \\ \vdots & \end{matrix}\right\}$$

The second type of feature is an existential feature for the relation. It also takes only two values $\{0, 1\}$ and when applied to an object, it takes the value 1 if the corresponding relation for that object is satisfied at all by any object. If such a feature exists, the system takes care to avoid redundancy of the representation of the property and relation value.

The Generic Model

The generic model does not describe an instance of the environment (a specific structure) but contains information on general rules that objects and structures in the environment will generally satisfy. The generic model is designed so as to allow direct use of the rules to compute properties and find objects which satisfy certain relations when this kind of information is requested by a user.

The basis of the knowledge representation in the generic model is, of course, the repertoire of: (1) classes of objects, (2) the available features, and (3) the relations. With each feature and relation, there is an associated algorithm which, when applied to an object of the adequate class, will come back with the value. These algorithms are constructed so that they may use only the resources (computer cycles) available to get an estimate of the value of the relation or parameter.

At the present, the rules are mostly man-made. Using a special interactive editor, experts generate and update rules expressed in the formats described below. It is anticipated that in future systems, the rules will be largely machine-generated by a rule learing subsystem.

Features

Each feature contains information defining the class of objects it operates on and the range of (integer) values that it can take.

There is an option to associate a name (an alpha-numeric string) with some or all of the integer values. These names are intended to facilitate man-machine interaction when properties are transmitted to or received from human operators. Similarly, the features' names are selected so that they will be self-explanatory as to their semantic meaning.

There are three types of features as determined by the type of the unique algorithm that is associated with the feature and is used to compute the properties of objects. The first type is programmed. Here, a preprogrammed routine is used to compute the value of the feature. Many of these properties are actually stored in the data structure of the object itself in which case they are not saved in the associative storage. These routines typically control directly the computer interfaced sensory instruments and get data from them with a minimum amount of analysis. The second type of features are those whose values are obtained from human operator or experts. When the value of one of those features for an object (the property) is required, the system issues a console (teletype or CRT) message requesting those values from the operator. The text of the message with blanks to be filled with the object names is stored in the data structure of the feature. The last type of features are those which are obtained by the inference system. With each such parameter, a unique classification tree is associated. This tree represents a sequential classification process. When a property of an object (the value of a feature) of that type is requested, the property is estimated by classifying the <u>object</u> into one of a few small categories of objects of that class, using other, hopefully simpler to obtain, properties. For each of the small categories, the generic model has a Bayesian estimate of the property for objects in that category. That estimate is then taken as the property of that object. The sequential classification can be easily edited to obtain finer categories and to update the Bayesian estimates of the property for objects in a category. This facilitates learning of the generic model. More detail on the classification tree data structure is given below.

Relations

The data structure of a relation specifies the class of objects it operates on and the class of objects on which it ranges. As is the case with parameters, there are three classes of relations as defined by the way they are computed. The first is the preprogrammed search where the search is done by a purely programmed algorithm. The second kind of relation requires that the list of objects will be provided by the operator. The third kind of relations, which we call inference relations, is computed from other (simpler) relations by union, intersection, and filtering the output of the simpler relations. The data structure of any relation contains pointers to the corresponding filter or existential features of that relation if they exist.

The rest of this article describes the representation of the sequential classification process associated with computation of inferred features and inferred relations.

Inferred Parameters – The Classification Tree

The essence of the inference process of properties (feature values) for objects is classification of the object into small categories where the range of values of a property for objects in the small category is very limited and, hence, the property can be estimated reliably.

Example: Consider three-dimensional bodies in an environment where three-dimensional bodies may be labeled only oranges, bananas, or table tops. Then, without any test, the *a priori* probability distribution of the label of an object of class three-dimensional body selected at random will be something like orange with probabability 0.6, table with probability 0.3, and banana with probability 0.1.

If we break the class of three-dimensional bodies into two categories, in the first category are those objects which have some planar surfaces, and in the second category are those objects which are purely curved surface. Then, the first category will include almost exclusively objects whose label takes the value tables, while the second category will be almost exclusively bananas or oranges. Testing the color and shape of objects in the second category will allow further subclassification of objects in that category into finer subcategories; some almost exclusively containing objects labeled bananas and the other of almost exclusively containing objects labeled oranges.

The classification tree for an inferred feature actually represents the classification process. The top node of the tree stands for the category of all objects of the class. With each node, there is an associated category of object and each son node stands for a (finer) subcategory of the category of objects in the parent node.

Each node contains the following information:

(1) Calling feature. Each node is part of a unique classification tree dedicated to one feature. A pointer of that feature is contained in the node.

(2) Default answer. This is the distribution of the values of the feature over all objects which belong to the category associated with the node. This Bayesian information is collected by going over examples and collecting the distribution. If the inference algorithm reaches the node without sufficient resources to expand it further, this estimate is returned as the answer. If there is not yet a known default estimate (insufficient training set), a marker to that effect is put in.

(3) List of sons and son selection procedures (optional). This information designates how to get a finer classification of the objects in the node's category if there are sufficient resources (compute time) to do so and get a finer estimate of the property. If there is no such list, the node is a terminal node (no finer classification is available for that case) and the default estimate is the only possible answer.

(4) Integration procedure. This specifies how to integrate estimates returning from the sons to get a unique answer returned from the current node if, because of ambiguity, more than one son is applied (that is, the object may have been estimated to belong with positive probability to more than one subcategory).

(5) Text (optional). Describing why the node was generated. This text is put in by the person or procedure who generated the node.

The son nodes correspond to the immediate subclassification of the objects in the category associated with the node. This finer classification is done based on other properties of the specific structure analyzed. Since these other properties may be nondeterministic, this selection of the subcategories may be nondeterministic. In such a case, the object will be assumed to belong to different sons (subcategories) with different positive probabilities, and the estimate of the answer is the average of the estimates obtained from the different possible subcategories weighted by the probability that the object belongs to each of those categories. The integration procedure specifies which kinds of averaging are used for the node.

The reader is reminded that the default answer is returned as the estimate of the property from a node if there is not sufficient compute time (resources) allocated to the expansion of the node so as to obtain finer subclassification (activate sons and average the estimates returned from those sons). Since all computations consume machine cycles (resources), the selection procedure is guaranteed to terminate after a finite amount of time.

The sons selection procedure in the nodes specifies a list of properties to be extracted from the structure analyzed. Each one of these properties is allocated a portion (specified in the node) of the resources still available at that point to the inference process for expanding the node. Each of these properties is estimated. A vector of property values is generated $\vec{P} = (P_1, P_2 \ldots P_N)$, where P_i $(1 \le i \le N)$ is the nondeterministic estimate for the corresponding property specified in the node. These properties may be properties of other objects designated by their relation to the current object. In this case the relations specified in the node need to be evaluated to find out to which objects those properties belong. The node contains information as to what portion of the available resources should be allocated to those searches, the type of call on the relations, and a special son (subcategory of objects) is specified for those cases where no object appears to satisfy the relation.

The vector of properti' ı is then reduced into a single property, like value Q, by an expression of the form

$$Q = \vec{P}^T \cdot A \cdot \vec{P} + \vec{B} \cdot \vec{P}$$

A is a square matrix N by N, and B a vector of N elements associated with the node Q is compared against a list of increasing thresholds $(T_1 \cdots T_m)$ $T_i < T_{i+1}$ and $T_m = +\infty$. With each threshold, a son node is associated such that if $T_{i+1} < Q \leq T_i$, then the ith son is selected (or in other words, the object belongs to the ith subcategory which is associated with the ith son).

Now since Q is a property like value, it can take a range of values with different probabilities, it is nondeterministic. As a result, the object may have positive probability of being in more than one subcategory, which will require integrating the estimates returned from each son node so as to get a unique answer from the parent node. There are a limited number of options for integration. Typically, it is geometrical or arithmetic averaging (weighted by the probabilities of belonging to the son) of the answers coming back from the different activated sons.

<u>Inferred Relation — Combination Searches</u>

Combination searches are organized in a search tree. The nodes of a search tree contain a son selection procedure which is used for resource allocation and expansion of the node. The selection procedure is identical to the son selection procedure used in the classification tree. Also each such node contains a pointer to the one relation with which it is associated. However, clearly there is no default answer associated with the node (the generic model cannot know the specific objects which will satisfy the relation for a specific object), and there is only one form of integration and that is to take the (weighted) union of the objects lists returned from the different activated sons.

The default answer of the feature tree is replaced at terminal nodes of the search tree by calls on other (hopefully simpler) relations and filters. Typically, at a terminal search node there will be specified a call on another relation and another filter feature. The object list returned from the other relation which is evaluated with a portion of the resources available for the expansion of that terminal node is then filtered by the feature which estimates the probability that each candidate actually satisfies the original relation and throws away all those which do not satisfy it with probability higher than 0.5.

Permanent Storage

The associative data base representing a specific structure is meaningless without the associated generic model which defines the classes of objects, the list of parameters, and list of relations. Hence, there are two types of permanent storage options on a magnetic disk or tape file. One is the generic model by itself, and the other is a generic model coupled with a description of a specific structure using the terms defined by the generic model. The system can store files containing either kind of data and accept them so as to continue analysis or editing from the status when saved.

Conclusive Remarks

The foregoing material describes our approach to the representation of perceptual information. We apologize for the lack of examples and incompleteness, which are mainly due to the fact that the complete system is not yet fully implemented, and as a result, not all points of ambiguity were resolved. But, we hope that the concepts of (1) inference and search bound by time constraints, (2) nondeterministic Bayesian inferences, and (3) nondeterministic description were shown to be of potential practical and integrated use for sensory data analysis systems. Most of the system described has been implemented but not yet applied to actual perceptual tasks. The actual recursive expansion algorithm that computes properties and values of relations, and the learning subsystem will be described in a later publication. We are also looking for practical ways to save the status of inference processes which exhausted their resources so that if additional resources become later available, the inference process may be resumed from the status it was terminated. Currently, it has to be started from the beginning again, but it can make use of any properties and values of relations that is already in the data base.

REFERENCES

1. N. Nilsson, "Problem Solving Methods in Artificial Intelligence," McGraw-Hill Book Co., Inc., New York, 1971.

2. M. O. Rabin, "Theoretical Impediments to Artificial Intelligence," Proceedings of I.F.I.P., 1974, pp. 615-619.

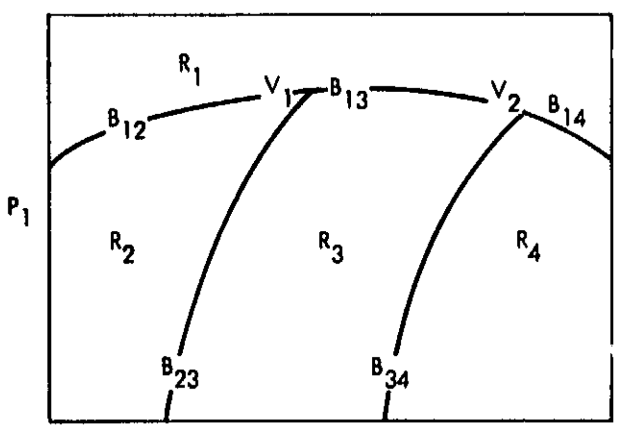
3. J. Feldman and Y. Yakimovsky, "Decision Theory and Artificial Intelligence: A Semantics Based Region Analyzer," A. I. Journal, 5 (1974), 349-371.

4. E. H. Shortliffe, "Mycin: A Rule Based Program for Advising Physicians Regarding Antimicrobial Therapy Selections," Stanford University Technical Report STAN-CS-74-465, 1974.

5. K. Van-Lehn, et al., "SAIL Manual," Stanford University Technical Report CS-STAN-73-373, July 1973.

6. J. Feldman and P. D. Rovner, "An Algol-Based Associate Language," C.A.C.M. Vol. 12, No. 8, 1969, pp. 439-449.

7. L. A. Zadeh, "Fuzzy Sets," Information and Control, Vol. 8, 334-353, 1965.



CLASSES OF OBJECTS: SCENES, REGIONS, BOUNDARIES, VERTICES

OBJECTS

OF CLASS	SCENES: P_1
OF CLASS	REGIONS: R_1, R_2, R_3, R_4
OF CLASS	BOUNDARIES: B_{12}, B_{13}, B_{14}, B_{23}, B_{34}
OF CLASS	VERTICES: V_1, V_2

FEATURES

OF REGIONS: SIZE, GRAY LEVEL, LABEL
OF BOUNDARIES: LENGTH, AVERAGE DIFFERENCES

RELATIONS

REGIONS IN:	SCENES $\longrightarrow$	REGIONS
BOUNDARY OF:	REGION $\longrightarrow$	BOUNDARY
ADJACENT:	REGION $\longrightarrow$	REGION
COMPOSES:	VERTEX $\longrightarrow$	BOUNDARY

PROPERTIES:

SIZE $\otimes$ $R_1 \equiv$ (10 1.0)

LABEL $\otimes$ $R_1 \equiv \begin{pmatrix} \text{"SKY"} & .5 \\ \text{"CLOUD"} & .5 \end{pmatrix}$

RELATION - OBJECT LIST:

$$\text{BOUNDARY OF} \otimes R_1 \equiv \begin{pmatrix} B_{12} & 1.0 \\ B_{13} & 1.0 \\ B_{14} & 1.0 \end{pmatrix}$$

$$\text{COMPOSES} \otimes V_1 \equiv \begin{pmatrix} B_{13} & 1.0 \\ B_{23} & 1.0 \\ B_{12} & 1.0 \end{pmatrix}$$

$$\text{REGIONS IN} \otimes P_1 \equiv \begin{pmatrix} R_1 & 1.0 \\ R_2 & 1.0 \\ R_3 & 1.0 \\ R_4 & 1.0 \end{pmatrix}$$

Fig. 1. Partial representation of a scene

NASA — JPL — Coml., L.A., Calif